(12) UK Patent Application (19) GB (11) 2 276 257 (13) A

(43) Date of A Publication 21.09.1994

(21) Application No 9405085.3

(22) Date of Filing 16.03.1994

(30) Priority Data
(31) 9300914 (32) 19.03.1993 (33) SE

(51) INT CL[5]
G06F 9/445

(52) UK CL (Edition M)
G4A AFL

(56) Documents Cited
EP 0419005 A2
IBM Technical disclosure bulletin, vol 22, No 10, March 1980 pages 4351-2, J A Aitken Jr.

(58) Field of Search
UK CL (Edition M) G4A AFL
INT CL[5] G06F 9/445

(71) Applicant(s)
ICL Systems AB

(Incorporated in Sweden)

Box 40, S-164 93, Kista, Sweden

(72) Inventor(s)
Hans Ingvar Berg
Gunnar Aija

(74) Agent and/or Address for Service
S M Dupuy
International Computers Limited, Cavendish Road, STEVENAGE, Hertfordshire, SG1 2DY, United Kingdom

(54) Configuration-sensitive program loading.

(57) A method for enabling the transfer of stored firmware during start-up of a computer, comprising the steps of reading configuration information stored in a configuration table in the first memory device (40), transferring selected software modules from the first memory device to a second memory device (130, 60, 50), and storing the selected software modules at selected address areas in the second memory device; in all accordance with the configuration information.

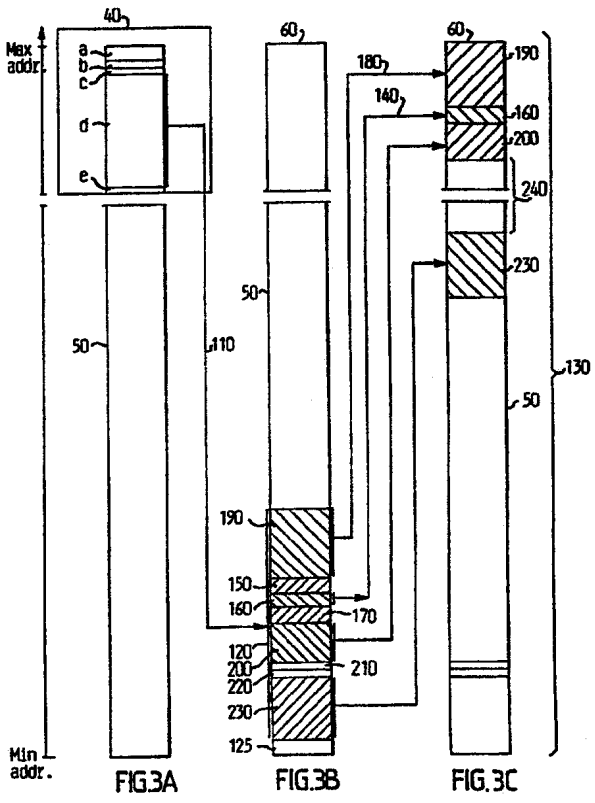

FIG.3A FIG.3B FIG.3C

GB 2 276 257

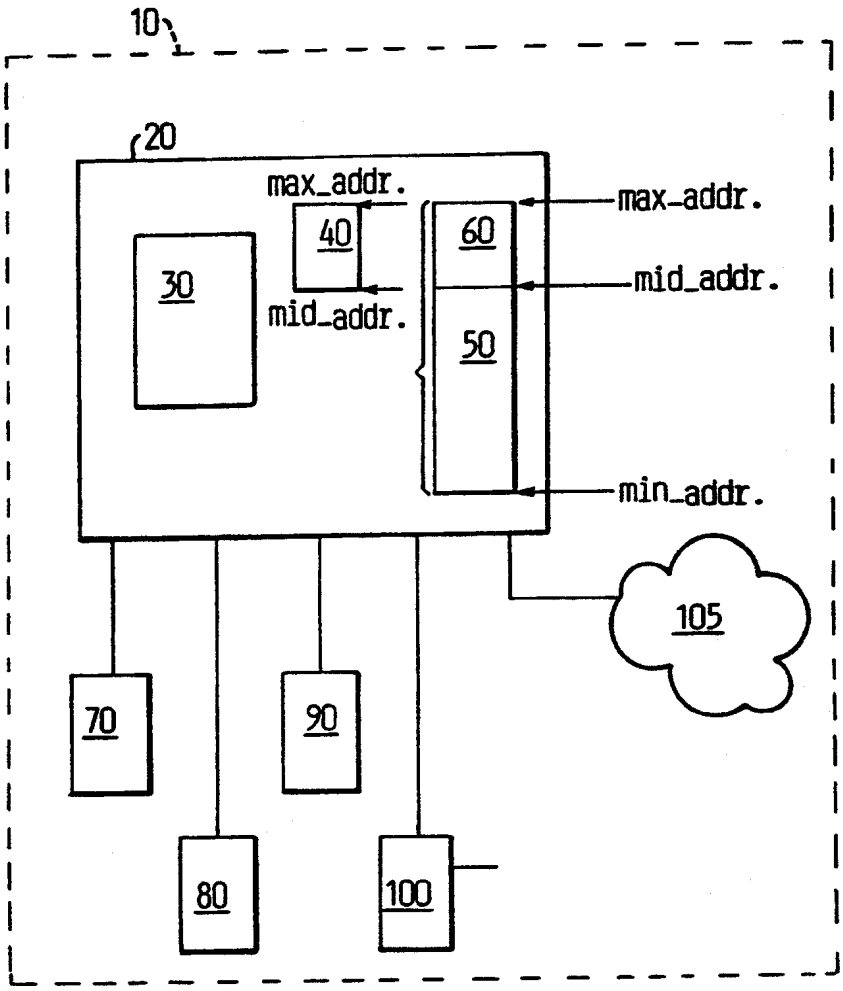
10
20
max_addr.
40
mid_addr.
30
60
50
max_addr.
mid_addr.
min_addr.
105
70
90
80
100

FIG.1

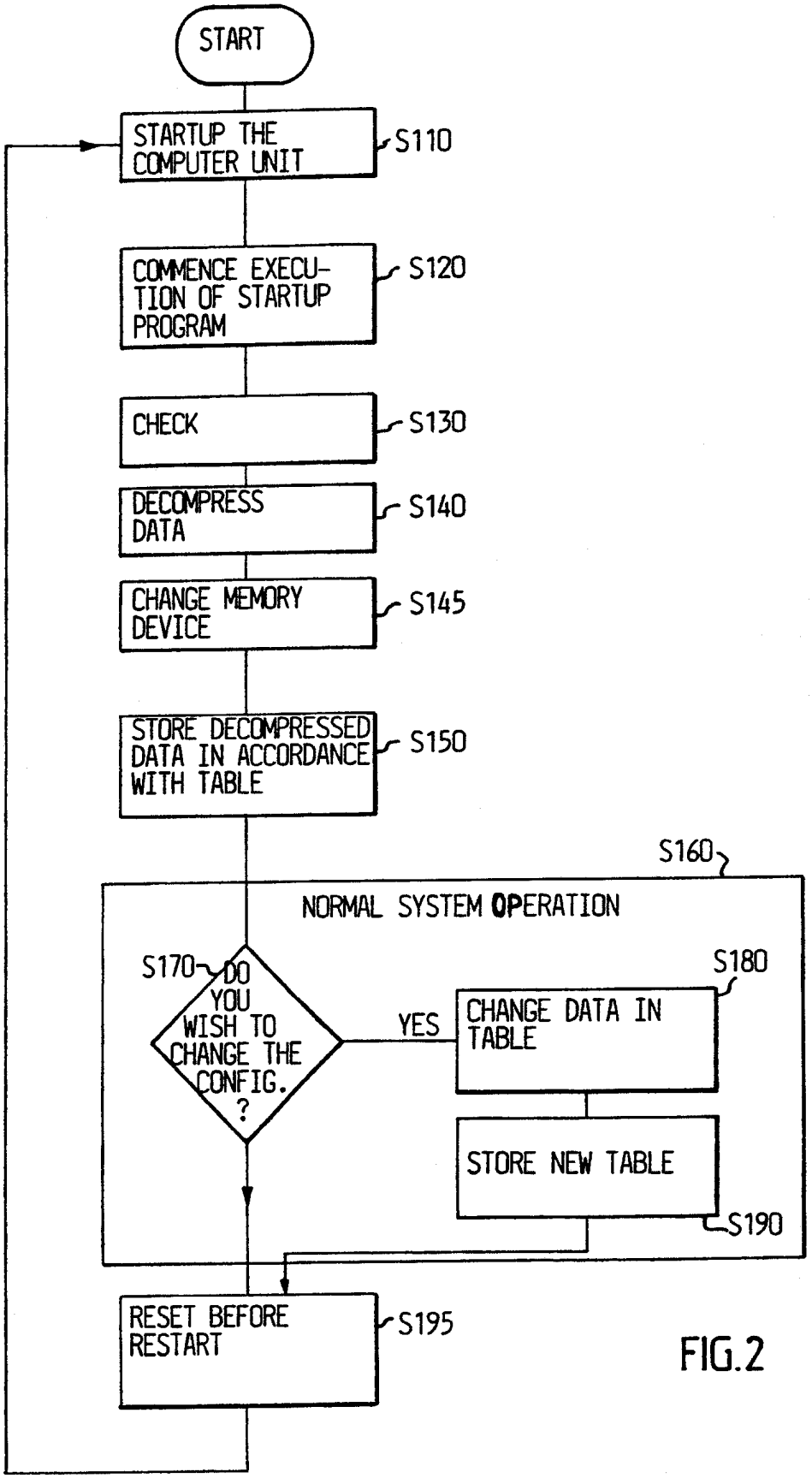
START
STARTUP THE COMPUTER UNIT
S110
COMMENCE EXECUTION OF STARTUP PROGRAM
S120
CHECK
S130
DECOMPRESS DATA
S140
CHANGE MEMORY DEVICE
S145
STORE DECOMPRESSED DATA IN ACCORDANCE WITH TABLE
S150
S160
NORMAL SYSTEM OPERATION
S170
DO YOU WISH TO CHANGE THE CONFIG. ?
YES
S180
CHANGE DATA IN TABLE
STORE NEW TABLE
S190
RESET BEFORE RESTART
S195

FIG.2

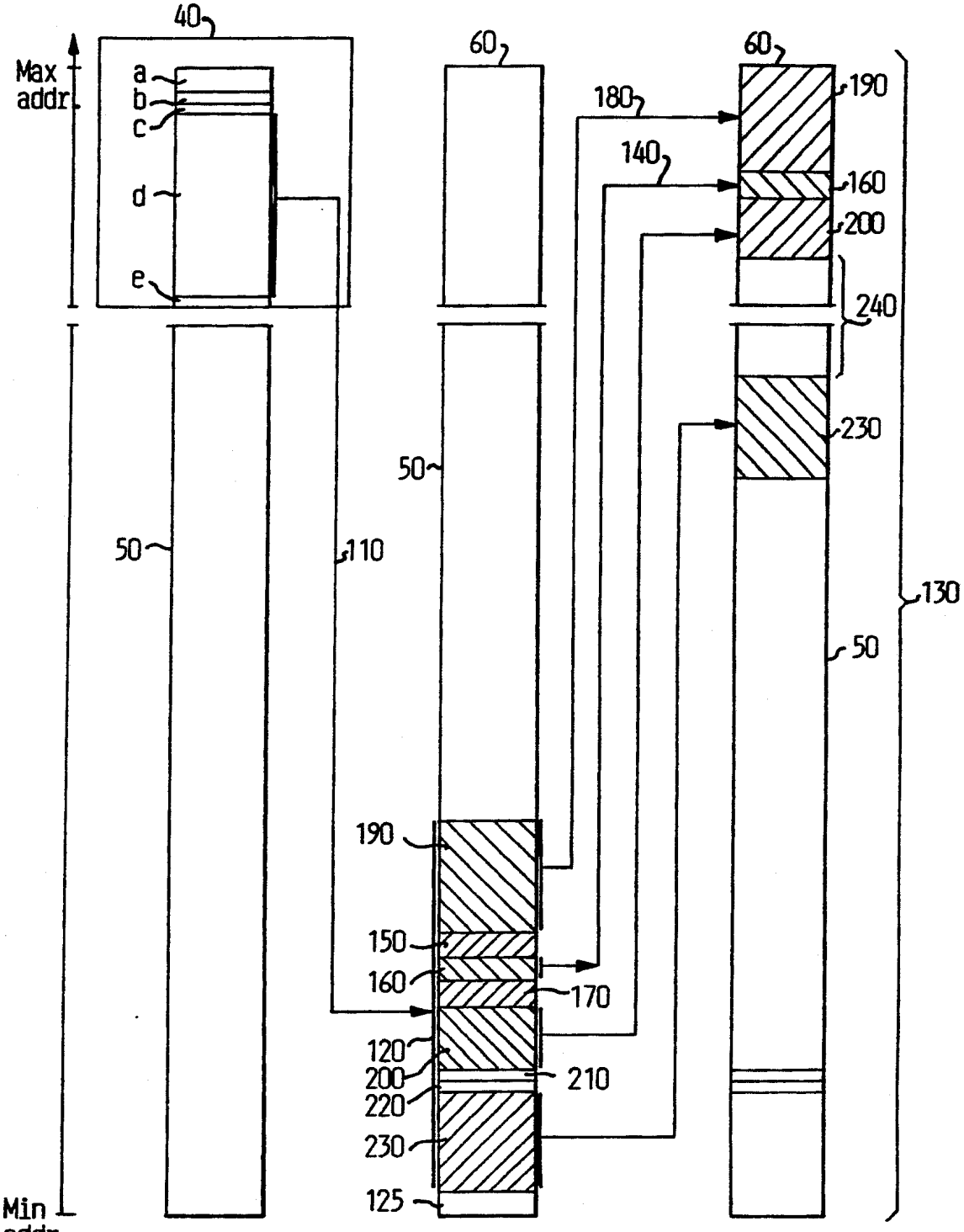

FIG.3A FIG.3B FIG.3C

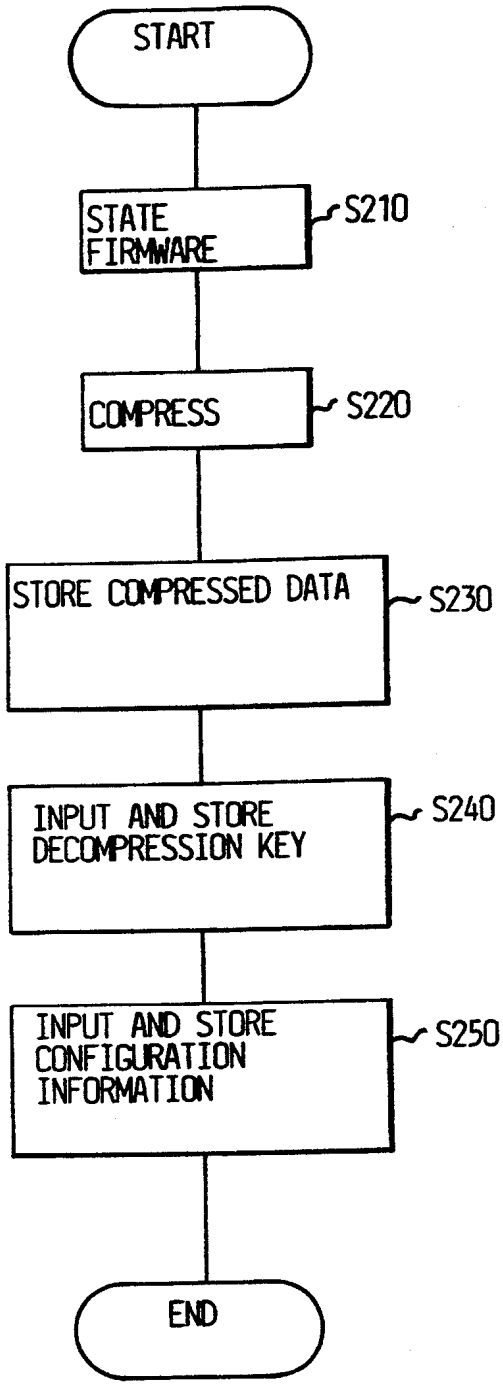
START
STATE FIRMWARE
S210
COMPRESS
S220
STORE COMPRESSED DATA
S230
INPUT AND STORE DECOMPRESSION KEY
S240
INPUT AND STORE CONFIGURATION INFORMATION
S250
END

FIG. 4

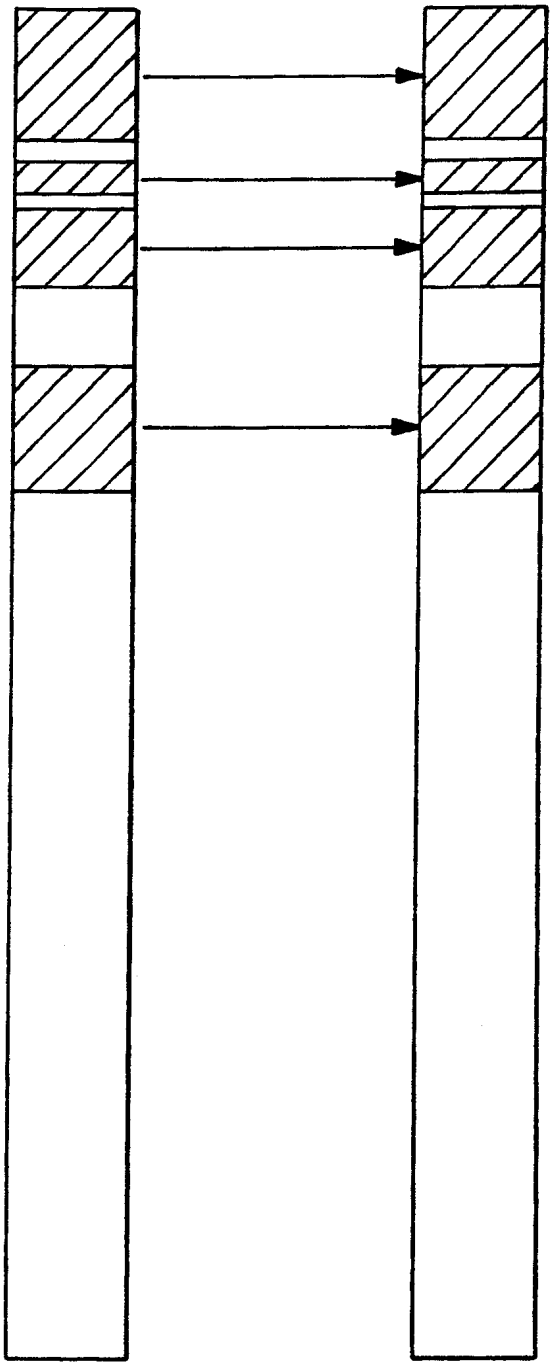

FIG.5A FIG.5B

2276257

A METHOD AND ARRANGEMENT RELATING TO COMPUTER SYSTEMS

This invention relates to a method for use when starting up a computer arrangement and also to an arrangement for carrying out the method.

The software used by a computer with each computer and/or computer system startup and restart routine is normally stored in a memory which retains its data even in the absence of a power supply, such as PROM-packages, for instance. A part of the software built into the computer system, often called Basic Input/Output Services or BIOS is typically divided into program modules, wherein only one module is stored in each PROM-package.

A program module will thus include functions which are executed in the event of a given type of interruption, such as interrupt type (10) hex for instance, which is concerned with a number of different video functions. The video function that the BIOS program is intended to perform in the event of the interrupt (10) hex is decided by the content of certain store registers.

In the case of certain specific computer system configurations, not all of the functions that the standard startup routine of the BIOS program can perform are required. Instead, there may be a need to utilize corresponding memory locations for other purposes.

When practising known techniques, it is necessary for a system administrator or user to change memory packages in the computer manually when wishing to change some part of the BIOS software of the computer system. Furthermore, when practising known techniques, it is not possible to utilize the memory of the computer system in an optimal fashion,

since the BIOS software is divided into a plurality of PROM-packages that have completely different memory addresses, each of the PROM-packages containing a program module. Each PROM-package has belonging thereto a number of jumpers which must be switched physically by the system administrator in order to select a start address for a corresponding BIOS program module.

In order to provide quicker access to the program modules, these modules are normally copied from the PROM-packages to a system-contained main memory which has shorter access times. By changing memories, part of the main memory will obtain the same addresses as the PROM-packages. When copying the program modules in accordance with known techniques, each individual program module is copied to the same address that it had in the PROM-package, although after being copied it is found in the main memory. This is illustrated by means of arrows in Fig. 5, while Fig. 5A illustrates the memory area which includes the PROM-packages and Fig. 5B illustrates the main memory. The hatched areas in Fig. 5A indicate software modules that are stored in PROM-packages. When practising the known technique, a separate PROM-package is required for each software module. Since the routine which deals with copying of the modules does not contain information concerning the functions of the different program modules, it is necessary to copy all modules. The only way in which the system administrator can avoid loading the main memory unnecessarily with program modules that are not required by the computer system is to physically (manually) remove corresponding PROM-packages or to block these packages with the aid of jumpers. There are several different reasons why it is impossible or extremely difficult to utilize the memory space in the PROM-units of the computer system when practising the known technique, among other things because different program modules occupy different volumes of memory space.

Given below is a list of terms used in the present document, and their significance:

Firmware	Identifies information or software stored in a permanent memory. Firmware may also be called firm software or built-in software.
Permanent Memory	A memory which will retain stored data even in the absence of power supply.
Volatile Memory	A memory which loses its stored data when the power supply is discontinued.
Work-Store	The storage of software in a memory area used when processing data. Work storing can be effected, for instance, in a work store.
Working Memory	A memory with which the data processing device primarily works.

It is an object of the present invention to provide a method which will enable firmware to be handled readily in a computer system.

Another object of the present invention is to enable the selection of which software modules shall be placed in which memory positions with the aid of software.

A further object of the present invention is to provide a method to enable program modules or data tables which are stored as firmware when starting-up the computer system to be repositioned such that the remaining unoccupied memory space can be readily utilized.

Still another object of the invention is to provide a method for reducing the need of permanent memory hardware in a computer system.

Yet another object of the invention is to enable the configuration of the system to be detected automatically and, in accordance therewith, adapt the choice of program modules and their location in the memory space.

A further object of the invention is to produce transferable stored firmware in a computer system arrangement.

Yet another object is to reduce the amount of hardware required by a computer system, for instance such hardware as address decoders, jumpers, permanent memory sockets, etc.

According to one aspect on the present invention there is provided a method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, characterized by the steps of reading changeable configuration information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information.

According to another aspect of the present invention there is provided a method for enabling stored firmware to be transferred in a computer system arrangement, wherein the arrangement includes at least one computer unit and at least

one first permanent memory device, characterized by storing in the permanent memory device a configuration table which includes information concerning the construction of the computer system with regard to hardware; storing in the permanent memory device software modules which are intended for use during operation of the computer unit; storing in the permanent memory device a startup program which, when executing, functions to transfer the information content of selected software modules to specific address areas in a working memory device in accordance with information disclosed in the configuration table.

According to a further aspect of the present invention there is provided a computer system arrangement which comprises at least one computer unit and at least one first memory device which includes at least one first data block in which information is stored in the form of firmware, wherein the computer unit is intended to execute a startup program stored in the first data block when starting-up the computer system, **characterized** in that the first memory device includes a second data block which includes software modules and a configuration table which includes address information; and in that, when starting-up the computer system, the computer unit is intended to read information from the configuration table and to transfer the information content of selected software modules to specific address areas in the second memory device in accordance with the address information read from the configuration table.

Embodiments of the invention will now be described with reference to the accompanying drawings, in which

Figure 1 illustrates schematically a computer system constructed in accordance with one embodiment of the invention;

Figure 2 is a flow diagram which illustrates one embodiment

of a method applied when starting-up a computer unit in accordance with the present invention;

Figures 3A, 3B and 3C are block schematics of memory areas in the memory devices of an inventive computer system;

Figure 4 is a flow diagram which illustrates an embodiment of a method applied when storing firmware in a computer unit in accordance with the invention; and

Figures 5A and 5B are block schematics which illustrate memory areas in memory devices of a computer system according to the known prior art.

Fig. 1 illustrates schematically a computer system 10 which includes a computer unit 20. The computer unit 20 may comprise a personal computer which includes a data processing device 30, for instance in the form of one or more microprocessors.

When starting-up the computer unit 20, the data processing device 30 reads information contained in a memory device 40. The memory means 40 may be a permanent memory. According to one embodiment of the invention, the permanent memory 40 is a FLASH-memory device. According to another embodiment of the invention, the permanent memory 40 is a conventional EEPROM. The memory device 40 may include firmware, such as startup software and BIOS program modules. According to the invention, certain software modules may be stored in a compressed form in the memory device 40. A compressed data packet which includes one or more software modules will take up less memory space than that taken up by a corresponding amount of data or information stored in an uncompressed state, as is well known to the person skilled in this art.

The data processing device 30 may utilize a memory area within given address limit values, where min_addr may

constitute the lowest address and max_addr can constitute the highest address.

The address value mid_addr can illustrate an address limit between memory devices that are available to the computer unit 20.

The data processing device 30 can avail itself of the memory device 40 precisely when starting-up the computer unit 20, when working with addresses within the range between mid_addr and max_addr, and can avail itself of a memory device 50 when working with data at addresses which lie within the range between min_addr and mid_addr. The memory device 50 may be a volatile memory.

After having completed given startup procedures, a point is reached in which the data processing device can "exchange" the memory device 40 for another memory device 60. Thus, after the break point, the data processing device 30 is able to reach the memory device 60 when addressing an address in the range mid_addr to max_addr instead of reaching the memory device 40.

The ability of being able to switch from one memory device 40 to another memory device 60 can afford the advantage of increasing the volume of memory area that the data processing device 30 is able to reach in a short access time. According to one embodiment of the invention, the memory device 60 is a volatile memory with short data access time.

One or more peripheral units may be connected to the computer unit 20. This is illustrated in Fig. 1 with the units 70, 80, 90 and 100. The computer unit 20 may also be connectable to a network 105, such as a Local Area Network, for instance, through which the computer unit 20 can communicate with other computer units.

Fig. 2 illustrates an embodiment of a first inventive method. The method illustrated in Fig. 2 makes it possible to choose which software modules shall be copied into the working memory, and also to choose the addresses in the working memory at which respective modules shall be stored, irrespective of the module addresses in the permanent memory device 40.

Step S110

As indicated by Step S110 in Fig. 2, the method illustrated in Fig. 2 can be initiated when starting-up or restarting the computer unit.

Steps S120 and S130

In Step S120, the data processing unit 30 included in the computer unit executes a first startup program. This startup program can be executed and when executed, the first startup program may include a routine which requires a check to be made as to which peripheral units are included in the hardware included in the computer system, as illustrated by Step S130 in Fig. 2. The resultant configuration data or information can be stored in a configuration table which will thus disclose those program modules which are required by the computer system in respect of the current configuration of said system.

According to another embodiment of the method, the configuration table can be changed by a user of the computer unit, and in the case of this embodiment Step S120 can be followed by Step S140.

It lies within the scope of the invention to change the configuration table both automatically, in accordance with Step S130, and manually, as described below. According to one embodiment of the invention, the configuration table can

be updated automatically with regard to a first group of software modules, whereas the configuration table is updated by means of interaction with a user with regard to a second group of software modules.

Step S140

In Step S140, the first startup program summons a decompression or decompaction program which unpacks or decompresses the firmware modules that are given in the configuration table and that are stored in a compressed state. According to another embodiment of the inventive method, the first decompression program decompresses all firmware modules stored in the memory device 40, wherein the choice of those modules that shall be used is made in accordance with the configuration table after said decompression.

Fig. 3A illustrates schematically the memory space in a computer unit memory. In the case of the Fig. 3A embodiment, the memory areas in **a**, **b**, **c**, **d** and **e** are memory areas which contain firmware, i.e. information which is not lost in the absence of power supply. This firmware is then found in the memory areas of the permanent memory device 40 (Fig. 1 and Fig. 3A).

The memory area **a** may include the program code that is executed when starting-up the computer unit, and the memory area **b** may be a general parameter area in which a number of parameters are stored.

The area **c** may include the aforesaid configuration table containing information as to which software modules shall be used, and information as to which addresses and software modules respectively shall be work-stored. The memory area **c** may also include the aforesaid decompression program, which can be used to decompress compressed data.

The memory area d may be a memory area which contains compressed software modules.

The memory area e may include a check table containing such information as first stored check-sums for the compressed modules and the decompression code. According to one embodiment of the invention, there is stored in the check table a check sum which applies to the decompression code and to the compressed modules. According to another embodiment of the invention, there is also stored a check sum which applies to the compressed software modules in the check table. Furthermore, the check table may include in the memory area e the memory address of the configuration table stored in the memory area c. The memory area e may also include such information as the date on which the configuration table stored in memory area c was last changed. The memory area e may also include information which discloses the number of times that firmware has been changed. This latter information finds particular importance when the firmware is stored in a memory which is only able to manage a given number of reprogrammings, such as in the case of some FLASH memories.

According to Fig. 2, there is calculated in Step S140 a second check sum which is based on all compressed software modules and which is compared with the check sum stored in the check table prior to decompressing the modules. When the check sums are found to agree, the startup program can read the configuration table stored in the memory area c and the firmware program can then decompress software and, as illustrated by the arrow 110 in Fig. 3, store the software in the memory device 50, in which access time is short. This storage in the memory device 50 may, however, have the form of an temporary storage so as then to "work-store" selected software modules, as described below.

In accordance with the method, an alarm function may be

activated when the check sums do not agree with one another.

Fig. 3A illustrates the memory area that is reached by the data processing device 30 when starting-up the computer unit. The arrow 110 illustrates that information **d** that has been stored in the memory device 40 in the form of firmware is decompressed and moved to the memory device 50.

Fig. 3B illustrates the memory area that is taken-up in the memory device 50 by the decompressed information **d**, in hatched areas 120.

Step S140 also involves copying the software modules **a**, **c** and **e** to a memory area 125 in the memory device 50 (Fig. 3B). This means that the executing program can copy itself and then continue to execute from the memory area 125.

According to one embodiment of the invention, the software modules in the memory areas **c** and **e** also include compressed data, such as the configuration table for instance, and consequently the aforedescribed decompression will also concern these modules, in accordance with this embodiment.

Step S145

The aforementioned break point can occur in Step S145, wherein the memory device 40 is exchanged for the memory device 60. The combination of the memory device 50 and the memory device 60 is called the working memory 130 (Fig. 3C) in the following. The working memory may be a main memory.

Step S150

In Step S150, the startup program can use the information in the configuration table to decide which of the program modules shall be used in the computer system and to obtain information as to where, i.e. at which addresses, respective

software modules shall be placed in the working memory 130 for future use, until the computer unit is stopped and restarted.

There is also effected in Step S150 a selection process in which only selected software modules are placed at those addresses in the working memory at which they shall be found available during normal operation of the computer system (Figs. 2 and 3C). Fig. 3B shows, among other things, three decompressed program modules 150, 160 and 170. The module 150 may be, for instance, an initiating routine for a Novel data server, whereas the module 160 may be a routine for a Unix server and the module 170 may be some other corresponding module which is only sometimes required. The arrow 140 illustrates an example in which only one of the modules 150, 160 or 170, namely the module 160, is selected and stored for use in the computer system.

In Step S150, this choice is effected with the aid of information contained in the configuration table. The configuration table may include a sub-table for each software module. Each individual sub-table is then related to an individual software module. A sub-table may include information in the form of a module flag, wherein each module flag can take either one of two different values, and wherein when the flag has a first value the corresponding module shall be work-stored in the working memory 130, as illustrated by the arrow 180 in Fig. 3 which shows that the software module 190 is moved or copied to an address area in the memory device 60. When the flag has a second value, the corresponding module shall not be used during operation of the system with the current configuration and the module is therefore not work-stored in the working memory. This latter refers, for instance, to the modules 150 and 170 in the example illustrated in Fig. 3. These two modules are left at their addresses according to Fig. 3B and will ultimately be typed-over with other information during operation of the

computer system.

The sub-table may also include module associated information which discloses the type of the corresponding software module. For instance, this information can disclose whether the module includes a program code or some other data, and/or information disclosing the type of program code.

The working memory 130 may have the configuration illustrated in Fig. 3C after having stored the software modules concerned in the working memory in accordance with the configuration table. Fig. 3C illustrates an example in which the software module 190 has been moved from its temporary storage in the memory device 50 to working storage in a first memory area in the memory device 60. Similarly, the module 160 has been moved to a second memory area in the memory device 60, this second memory area lying close to or contiguous with the first memory area with regard to address. The address areas used for work storing the software modules in the working memory can be write-protected to prevent erasure of the modules or typeover of the modules during operation of the system.

According to Step S140, decompression of the memory area d also resulted in software modules 200, 210, 220 and 230 which were stored temporarily in the area 120 of the memory device 50.

The software modules 210 and 220 may be modules which are used only prior to loading operative systems to the computer unit. According to the embodiment of the invention illustrated in Figs. 3B and 3C, these modules are retained at those memory addresses at which they were first stored in the working memory and these address areas need not be write-protected.

The module 200 can be placed in the address area beneath the

second memory area with the module 160 and bordering thereon.

According to one embodiment of the invention, the module 230 is placed so as to obtain a given address space 240 between the module 200 and the module 230. The address area 240 may be used, for instance, for a connectable memory means, such as a PCMCIA card, for instance.

Step S160

When Step S150 is completed, the computer unit can pass to normal operation, as illustrated with Step S160 in Fig. 2.

Steps S170, 180 and 190

During normal operation, the user or operator may be provided the possibility of initiating a change in the configuration table, as indicated with Step S170 in Fig. 2. This can be achieved, for instance, with a special reconfiguration command which provides access to the configuration table stored in the memory device 40. According to this embodiment, when the configuration table is stored in a compressed state in the memory device 40, the reconfiguration command can provide access to the decompressed configuration table in the working memory 130. According to this latter embodiment of the invention, the user is able to initiate compression and storage of the modified table in the permanent memory device 40.

In Step 180, the operator is able to store address information in the configuration table so as to enable the software modules to be work-stored in a manner which will enable the working memory to be utilized in an optimal fashion.

The operator is also able to insert information concerning those software modules that can be stored and write-protected

in the working memory.

Step S195

When the receiver does not request processing of the configuration information, Step S150 is followed solely by normal system operation until the computer unit is switched-off or reset as indicated in Step S195.

According to one embodiment of the invention, the firmware modules are stored in an uncompressed state in the memory device 40, and no decompression is carried out in this embodiment.

The invention also relates to a method of storing firmware in a computer unit so that software modules stored as firmware can be easily made movable to selected addresses in the working memory 130. Furthermore, the invention enables the memory space of a computer unit used to store firmware to be considerably reduced. This second inventive method is illustrated in Fig. 4 and described in detail herebelow.

Step S210

The following procedures are taken when storing the prospective inbuilt software in the computer unit: The method is commenced and the prospective inbuilt software, also called firm software, to be loaded into the permanent memory device of the computer unit is given in a first Step S210.

Steps S220 and S230

The selected software is compressed in Step S220 and is stored in Step S230 in a selected memory area in the first memory device 40, which will retain its data content in the absence of an applied voltage.

Step S240

In Step S240, there is inserted a decompression program, which can also be stored in the first memory device 40.

Step S250

According to one embodiment of the invention, there is inserted in Step S250 configuration information which may also be stored in the first memory device. The configuration information may have the form of a data table which discloses, among other things, the memory addresses at which the various software modules shall be stored when decompressed. The table may also include information as to which software modules shall be work-stored in the working memory and/or the addresses at which the modules shall be work-stored when carrying out the first method according to the invention, as described above.

According to one variant of the second method, all the respective inbuilt software, including BIOS software, configuration table and other software modules is compiled before all this information is stored, essentially simultaneously, in the permanent memory device 40. According to this inventive method, selected modules are compressed before making this compilation so that the data packet which is later loaded into the permanent memory device will have precisely the content that one wished to store in the permanent memory device 40.

Since, in accordance with the invention, a large part of the firmware can be stored in a compressed state, the need for memory space in the permanent memory device is correspondingly smaller. This enables the amount of hardware to be reduced for the same amount of firmware. According to one embodiment of the invention, all firmware is stored in one and the same FLASH-memory package.

According to another embodiment of the invention, the software modules are stored in the permanent memory device 40 in an uncompressed state, together with the aforesaid configuration information and startup program.

CLAIMS

1. A method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, **characterized** by the steps of reading changeable configuration information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information.

2. A method according to Claim 1, **characterized** by decompressing at least one second data block stored as compressed firmware in the first memory device when starting-up the computer unit, and storing the decompressed data block in the second memory device.

3. A method according to Claim 1 or 2, **characterized** by decompressing software, such as a BIOS program, included in the second data block.

4. A method according to Claim 1 or 2, **characterized** by decompressing the configuration information included in the second data block.

5. A method according to Claim 2, 3 or 4, **characterized** by temporarily storing the decompressed data block in the second memory device and selecting the software modules in the decompressed data block that shall be work-stored in the second memory device, in accordance with

configuration data.

6. A method according to any one of the preceding Claims, **characterized** by detecting the configuration of the computer system, such as hardware components included in the system, and updating the configuration information included in the configuration table with the aid of a computer program included in the computer system and on the basis of the knowledge of the system configuration.

7. A method according to any one of the preceding Claims, **characterized** by placing each software module on an address area intended for respective software modules in accordance with address information included in the configuration table, i.e. to work-store the selected software modules, and write-protecting those addresses at which the selected software modules are worked-stored.

8. A method according to any one of the preceding Claims, **characterized** by calculating a first check sum on the basis of the compressed data block; comparing the first check sum with a second check sum stored in the permanent memory device and decompressing the data block when agreement is found between the thus compared check sums, and activating an alarm function when disagreement is found between the check sums.

9. A method for enabling stored firmware to be transferred in a computer system arrangement, wherein the arrangement includes at least one computer unit and at least one first permanent memory device, characterized by storing in the permanent memory device a configuration table which includes information concerning the construction of the computer system with regard to hardware; storing in the permanent memory device (40) software modules which are intended for use

during operation of the computer unit; storing in the permanent memory device a startup program which, when executing, functions to transfer the information content of selected software modules to specific address areas in a working memory device in accordance with information disclosed in the configuration table.

10. A method according to Claim 9, **characterized** by storing compressed software modules in the permanent memory device; storing in the permanent memory device a decompression program which, when executing, functions to decompress compressed software modules; arranging the decompression program so that it can be called by the startup program; and storing in the permanent memory device a module address list which includes information disclosing the address at which each software module shall be stored in the working memory.

11. A computer system arrangement which comprises at least one computer unit and at least one first memory device which includes at least one first data block in which information is stored in the form of firmware, wherein the computer unit is intended to execute a startup program stored in the first data block when starting-up the computer system, **characterized** in that the first memory device includes a second data block which includes software modules and a configuration table which includes address information; and in that, when starting-up the computer system, the computer unit is intended to read information from the configuration table and to transfer the information content of selected software modules to specific address areas in the second memory device in accordance with the address information read from the configuration table.

12. A computer system arrangement according to Claim 11, **characterized** in that the second data block includes

compressed software modules; in that the first data block includes a data decompression program; and in that the startup program is constructed to summon the decompression program and therewith cause the compressed software modules to be decompressed.

13. A computer system arrangement according to Claim 11 or 12, **characterized** in that the first memory device is a permanent memory device; and in that the second memory device is a working memory device which belongs to the computer unit.

14. A method of use when starting up a computer system arrangement substantially as herein described with reference to Figs 1-4 of the accompanying drawings.

15. A computer system arrangement substantially as herein described with reference to Figs 1-4 of the accompanying drawings.

Patents Act 1977 Examiner's report to the Comptroller under Section 17 (The Search report) -22-	Application number GB 9405085.3
Relevant Technical Fields (i) UK Cl (Ed.M) G4A AFL (ii) Int Cl (Ed.5) G06F 9/445	Search Examiner MR S J PROBERT Date of completion of Search 25 MAY 1994
Databases (see below) (i) UK Patent Office collections of GB, EP, WO and US patent specifications. (ii)	Documents considered relevant following a search in respect of Claims :- 1-15

Categories of documents

X: Document indicating lack of novelty or of inventive step.

Y: Document indicating lack of inventive step if combined with one or more other documents of the same category.

A: Document indicating technological background and/or state of the art.

P: Document published on or after the declared priority date but before the filing date of the present application.

E: Patent document published on or after, but with priority date earlier than, the filing date of the present application.

&: Member of the same patent family; corresponding document.

Category	Identity of document and relevant passages	Relevant to claim(s)
X	EP 0419005 A2 (IBM) See whole document	1, 9 and 11 at least
X	IBM Technical Disclosure Bulletin, Vol 22, No 10, March 1980 "Configuration-sensitive program load", pages 4351-2, J A Aitken Jr	1-15

Databases: The UK Patent Office database comprises classified collections of GB, EP, WO and US patent specifications as outlined periodically in the Official Journal (Patents). The on-line databases considered for search are also listed periodically in the Official Journal (Patents).